APPENDIX I - SYSTEM HANDLER REGISTRAR

```
import java.lang.*;
import java.security.*;

/**
 * Registration mechanism for trusted applications to set the error handler.
 */
public class SysHandlerRegistrar
{
  public final static int ERROR_INFO_EVENT_HANDLER = 0x0;
  public final static int REBOOT_EVENT_HANDLER = 0x1;
  public final static int RESOURCE_DEPLETION_EVENT_HANDLER = 0x2;

  private static SysHandlerRegistrar theSHR = null;
  private static IEventHandler theErrorInfoEventHandler = null;
  private static IEventHandler theRebootEventHandler = null;
  private static IEventHandler theResourceDepletionHandler = null;

  /**
   * Zero argument constructor is protected so an application cannot create it.
   */
  protected SysHandlerRegistrar()
  {
  }

  /**
   * Get the singleton instance of the system handler registrar.
   *
   * @param type - ERROR_INFO_EVENT_HANDLER, REBOOT_EVENT_HANDLER, or
   * RESOURCE_DEPLETION_HANDLER.
   *
   * @return The system handler registrar.
   */
  public static SysHandlerRegistrar getInstance()
  {
    if(theSHR == null)
      theSHR = new SysHandlerRegistrar();

    return theSHR;
  }

  /**
   * Get the system system handler.
   *
   * @param type - ERROR_INFO_EVENT_HANDLER, REBOOT_EVENT_HANDLER, or
   * RESOURCE_DEPLETION_HANDLER.
   *
   * @return Currently registered handler for the type specified.
   *
   * @throws SysHandlerPermission if the application does not have permission
   * to get the handler.
   */
  public static IEventHandler getEventHandler(int type) throws SecurityException
  {
    SysHandlerPermission ehp = new SysHandlerPermission("getEventHandler");
    // If the caller does not have permission the AccessController will throw a
    // SecurityException.
    //AccessController.checkPermission(ehp);    to be uncommented

    if(type == ERROR_INFO_EVENT_HANDLER)
      return theErrorInfoEventHandler;
    else
      if(type == REBOOT_EVENT_HANDLER)
        return theRebootEventHandler;
      else
        return theResourceDepletionHandler;
  }

  /**
   * Set the system event handler.
   *
   * @param type - ERROR_INFO_EVENT_HANDLER, REBOOT_EVENT_HANDLER, or
   * RESOURCE_DEPLETION_HANDLER.
   *
   * @param seh - System event handler created by the registering application.
   *
   * @throws EventHandlerPermission if the application does not have permission
   * to set the handler.
   */
  public static void setEventHandler(int type, IEventHandler seh) throws SecurityException
  {
    SysHandlerPermission ehp = new SysHandlerPermission("setEventHandler");

    // If the caller does not have permission the AccessController will throw a
    // SecurityException.
    //AccessController.checkPermission(ehp);    to be uncommented

    if(type == ERROR_INFO_EVENT_HANDLER)
      theErrorInfoEventHandler = seh;
    else
      if(type == REBOOT_EVENT_HANDLER)
        theRebootEventHandler = seh;
      else
        theResourceDepletionHandler = seh;

  }

  /**
   * Unset the system event handler.
   *
   * @param type - ERROR_INFO_EVENT_HANDLER, REBOOT_EVENT_HANDLER, or
   * RESOURCE_DEPLETION_HANDLER.
   *
   * @throws EventHandlerPermission if the application does not have permission
   * to unset the handler.
   */
  public static void unsetEventHandler(int type) throws SecurityException
  {
    SysHandlerPermission ehp = new SysHandlerPermission("setEventHandler");

    // If the caller does not have permission the AccessController will throw a
```

```
        // SecurityException.
        //AccessController.checkPermission(ehp);     to be uncommented

        if(type == ERROR_INFO_EVENT_HANDLER)
          theErrorInfoEventHandler = null;
        else
          if(type == REBOOT_EVENT_HANDLER)
            theRebootEventHandler = null;
          else
            theResourceDepletionHandler = null;

      }

    }
```